THE DYNAMIC CIPHERS – NEW CONCEPT OF LONG-TERM CONTENT PROTECTING

Dr. Grzegorz Szewczyk, Central Ostrobothnia University of Applied Sciences
Department of Business And Technology, Chair of Information Systems
Kokkola, Finland, grzegorz.szewczyk@cou.fi

ABSTRACT: In the paper the original concept of a new cipher, targeted at this moment for civil applications in technology (e.g. measurement and control systems) and business (e.g. content protecting, knowledge-based companies or long-term archiving systems) is presented. The idea of the cipher is based on one-time pads and linear feedback shift registers. The rapidly changing hardware and software environment of cryptographic systems has been taken into account during the construction of the cipher. The main idea of this work is to create a cryptosystem that can protect content or data for a long time, even more than one hundred years. The proposed algorithm can also simulate a stream cipher which makes it possible to apply it in digital signal processing systems such as those within audio and video delivery or telecommunication.

Keywords: Content protection, Cryptosystem, Dynamic cryptography, Linear Feedback Shift Registers, Object-oriented programming, One-time pad, Random key, random number generators, Statistical evaluation of ciphers.

JEL Codes: D83.

1. Introduction

Many companies need to keep data that are very important for them (for example, research results, technological parameters, lists of customers, other content) secret for a long time; also, data stored on electronic media (hard disks, CD ROMs). This problem especially concerns organisations such as knowledge-based companies, travel agencies, and health-care institutions, as well as companies that want to be operators of e-commerce including content and entertainment delivery.

Long-term content protection and archives are today a major challenge for information technology. Rapidly changing hardware and software environments lead to the situation that information ciphered today can be cracked tomorrow because of new, faster computers, or embodied into software public knowledge and experiences of hackers. This problem particularly concerns content and sensitive personal data (e.g. property or identification). The best protection of content or archived data is to make cracking unfeasible. In practice this means that hardware and software employed for cracking would be much more worth than the information intended to be cracked. We also have to remember that content protection is related to copyright. So, good protection of content keeps its authors' rights, too.

To decrypt effectively stored data, two facts must be known: (1) the decoding algorithm and (2) the cryptographic key. Modern cryptosystems must be and actually are based on Kerckhoffs' law: *A cryptosystem should be secure even if everything about the system, except the key, is public knowledge* (1) (2). ERIC RAYMOND (3) extends this principle to the support of open source software, saying: *Any security software design that does not assume the enemy possesses the source code is already untrustworthy; therefore, "never trust closed source".* This results in the conclusion that if we want to have a secure cryptosystem for long-lasting content and data protection, we have to assume that a hacker knows or at least guesses the algorithm; moreover, he/she can have access to the open source code being used. However, this needs some cooperation of an insider, but all that we

have to do is to be aware that today's attacks on stored data come rather from the inside of a company than outside. In addition, social engineering plays a very important role in the disclosure of a cryptosystem. An interesting discussion on data security threats can be found, for instance, in (4).

Cryptosystems based on symmetric, block algorithms are commonly used for ciphering data for long-lasting storage. Block based algorithms do encoding using the same key for each part of information (a block). Taking into account that today pieces of information are large (pictures, technical drawings, audio, video) there is always enough data for cryptanalysis. The security of such cryptosystems depends on the length of the cryptographic key. LENSTRA & VERHEUL (5) researched the size of the key. They assumed that the size of the cryptographic key depends primarily on:

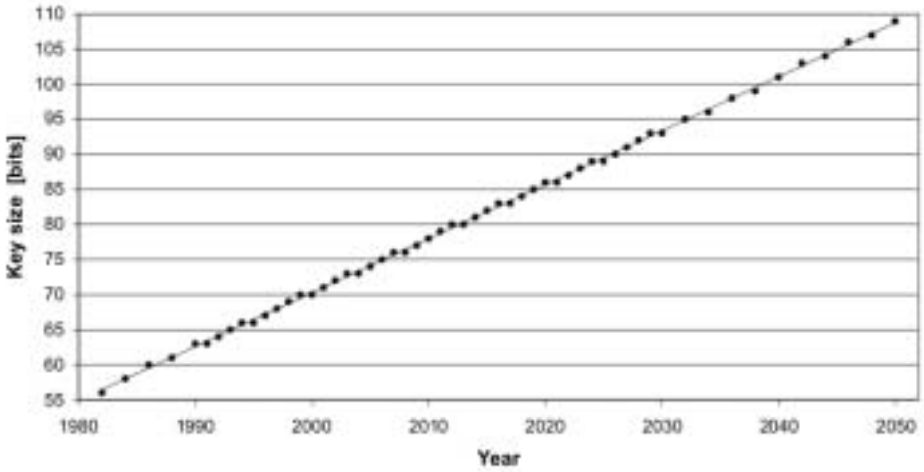

Fig. no. 1. - Secure size of the cryptographic key (block, symmetric ciphers) (5).

1. Life span: the expected time the information needs to be protected.
2. Security margin: an acceptable degree of infeasibility of a successful attack.
3. Computing environment: the expected change in computational resources available to attackers.
4. Cryptanalysis: the expected developments in cryptanalysis.

All considerations have been made from Data Encryption Standard (DES) point of view what doesn't exclude commonalty of conclusions. Authors assumed that the only reasonable and effective attack is the key space exhausting one ("brute-force attack"). Fig. no. ***1*** presents results of these researches. Information found in Fig. no. ***1*** can be interpreted in the following way: if we assume that information should be secure for next 25 years from year 2008, meaning by year 2033, a key of at least 95 bits long should be selected.

Table no. 1.

Average time estimated for a hardware brute-force attack (6)[1].

Hardware Cost [USD]	LENGTH OF KEY IN BITS					
	40	56	64	80	112	128
10^{4}	22,10 [ms]	23:12,116 [min]	4:00:51:05,501 [days]	773,4 [years]	10^{14} [years]	10^{17} [years]
10^{5}	2,21 [ms]	02:19,212 [min]	09:48:39,982 [hrs]	77,3 [years]	10^{13} [years]	10^{16} [years]
10^{6}	0,22 [ms]	00:13,921 [min]	01:03:38,376 [hrs]	7,7 [years]	10^{12} [years]	10^{15} [years]
10^{7}	0,02 [ms]	00:01,392 [min]	05:57,973 [min]	282,5 [days]	10^{11} [years]	10^{14} [years]
10^{8}	2,21 [µs]	00:00,139 [min]	39,775 [s]	28,2 [days]	10^{10} [years]	10^{13} [years]
10^{9}	0,22 [µs]	13,921 [ms]	3,580 [s]	2,7 [days]	10^{9} [years]	10^{12} [years]
10^{10}	0,02 [µs]	1,392 [ms]	0,354 [s]	6,4 [hrs]	10^{8} [years]	10^{11} [years]
10^{11}	0,00 [µs]	0,144 [ms]	0,033 [s]	38,2 [min]	10^{7} [years]	10^{10} [years]
10^{12}	0,00 [µs]	0,011 [ms]	0,003 [s]	4,0 [min]	10^{6} [years]	10^{9} [years]

Table no. 1 shows how long the key should be in order to protect information of certain value (the cost of the breaking cipher should be less or equal to the value of protected information). In years following 2008, the time presented in this table might be less because of Moore's Law.

1.1 One-time pad and dynamic cryptography

The only secure cipher is the one-time pad. The term "one-time pad" refers to any method of encryption where each byte of the plaintext is enciphered using one byte of the key stream, and each key byte is used one time, and then never used again. The key stream for a one-time pad must be a true-random stream, meaning that every key byte can take any of the values 0 to 255 with equal likelihood, and independently of the values of all other key bytes. One-time pad encoding can be usually denoted as

$$C_i = P_i \oplus K_i$$

where P_i (i = 0, 1, 2, …) is the i-th character of the plain text, K_i is the i-th byte of the key used for this particular message, and C_i is the i-th character of the resulting cipher. XOR is the most common operation used in one-time pad ciphering, but it can be replaced by any other. One-time pad decoding is actually a similar operation to the encoding (7) and has the following form:

$$P_i = C_i \oplus K_i$$

One-time pads were widely used by Soviet Intelligence (GRU) during WWII. There still exist messages that remained secret even today because the respective keys are missing. There is no help to use contemporary methods and tools of cryptanalysis for decoding. This information remains secret forever (8). It proves the strongest of one-time pads.

[1] Original data in (6) are presented for year 1995. Data in above table reflect situation in year 2008.

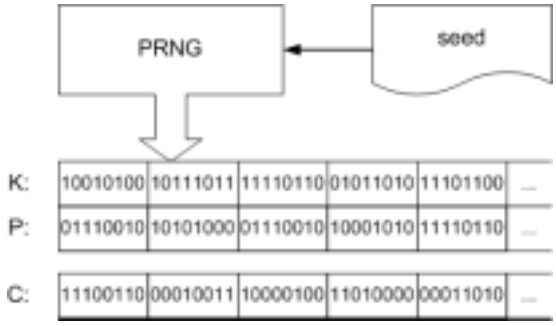

Fig. no. 2 - The conception of dynamic cryptography.

The practical difficulty of using a one-time pad is that the key bytes cannot be reused. This means that even for a two-way exchange of messages, each part must have a sufficient supply of key material at hand so that they cannot run out before more key can be furnished. Keys management is also very difficult. Large pieces of information in electronic or traditional form have to be moved between users which makes such a cryptosystem vulnerable (7).

Looking for practical implementation of one-time pads MARTINEZ (9), for software based solutions, and RITTER (10), for hardware based solution, proposed new encryption technology called dynamic cryptography. The idea of this technology is presented on Fig. no. **2**. A pseudo-random numbers generator (PRNG) generates an infinitive sequence of bits. This sequence is used as a one-time pad for encoding and decoding information. In a sequence of pseudo-random numbers, the value of each byte after the first several is mathematically derived from the values of a few preceding bytes. Seed (initial sequence) is a secret value that initialises the PRNG. An interesting discussion on problems related to design of cryptographically save PRNG is found in (11) (12).

1.2 Problem

Taking into account the current state of the art in the described area, decisions have been made to develop original dynamic ciphering algorithm targeted at this moment for civil applications in areas such as research, technology and business administration. It is assumed that the new cipher must be able to operate on blocks of variable-length and that Kerckhoffs' law (1) (2) is to be kept in relation to the class of algorithms only.

The aims of this paper are to present the original conception of the cipher, to open public discussion, and, finally, to make it possible for the community to carry out its own tests and cryptanalysis.

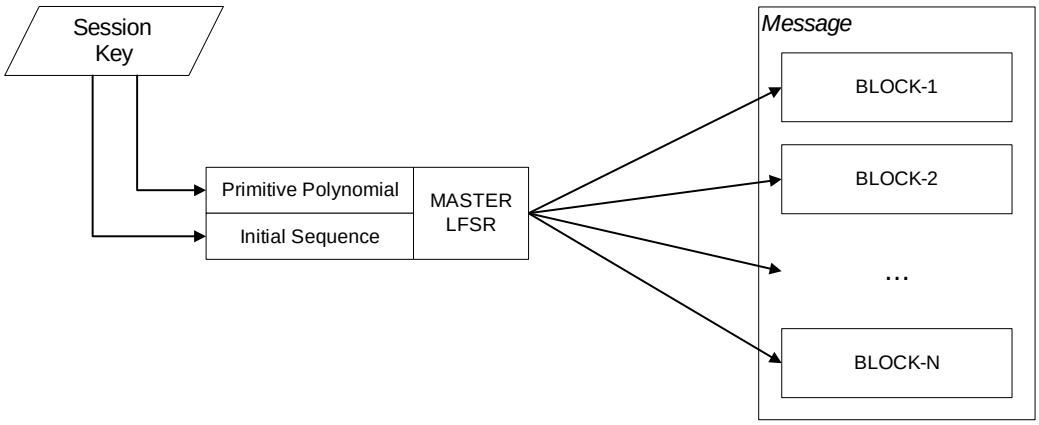

Fig. no. 3 - General conception of the KARHU44 ciphering algorithm.

2.Algorithm

The general idea of the proposed KARHU44 algorithm is presented in Fig. no. **3**. It is based on the conception of the one-time pad presented in section 1.1. Both formulas found in this section are used for encoding and decoding respectively. Pseudo-random numbers generators are made based on Linear Feedback Shift Registers (see section 2.1 for details).

To reach high linear complexity of the cipher it was assumed that each encoding / decoding round is using an exclusive ciphering sequence of pseudo-random numbers. The encoding / decoding round means either ciphering of different blocks or repeated ciphering of the same block. The exclusivity of the ciphering sequence results from the configuration of the LFSR (length and tap sequence) selected at random from the list and from the initial state of the LFSR (cryptographic key – see section 2.2 for details) generated for each round separately. Further, each block is encoded sixteen times in a row which ensures a sufficient level of information dissolving.

According to (6) any ciphering algorithm should contain two general ciphering techniques: substitution and transposition. The substitution is introduced to KARHU44 by means of the XOR operation on data and an infinitive sequence of bits generated by relevant LFSR.

The transposition technique is implemented in the algorithm by means of bit-mixing operation that is carried out before and after all XOR operations (see Fig. no. **4**). Each mixing operation uses own mixing table that is created at random. The source of randomness for mixing is a generator created in the same way like for XOR rounds (LFSR of a round).

The main role in this algorithm is played by the Master LFSR. It is the source of random numbers for selecting the configuration of other generators and it simulates the time-related element in the procedure of generating the cryptographic key. The length of the Master LFSR and its tap sequence is obtained from the session key. However, we should pay attention to the fact that this generator does not have a direct impact on the ciphering sequence because it acts like a control of all generators used for ciphering (mixing and xor-ing).

It was assumed that the length of the register of Master LFSR must be at least 160 bits long. Over the initial sequence session, the key also contains a tap sequence. Construction and generation methods are described in (13).

After the Master LFSR is initialised based on the session key, the list of LFSR configurations is permutated at random according to the algorithm presented in (14). Then for each round of encoding / decoding, the configuration of LFSR can be selected and a round key generated. Now, the LFSR for the current round is ready to use in the ciphering procedure.

2.1 Pseudo-random number generator

Generators of pseudo-random numbers used in this algorithm are made based on linear feedback shift registers (LFSR) having Fibonacci configuration. Shift register sequences are used in cryptography for a long time. There is a wealth of theory about them; streams ciphers based on shift registers have been the workhorse of military cryptography since the beginning of electronics. Ciphers made up of shift registers can be easily implemented in digital hardware (6).

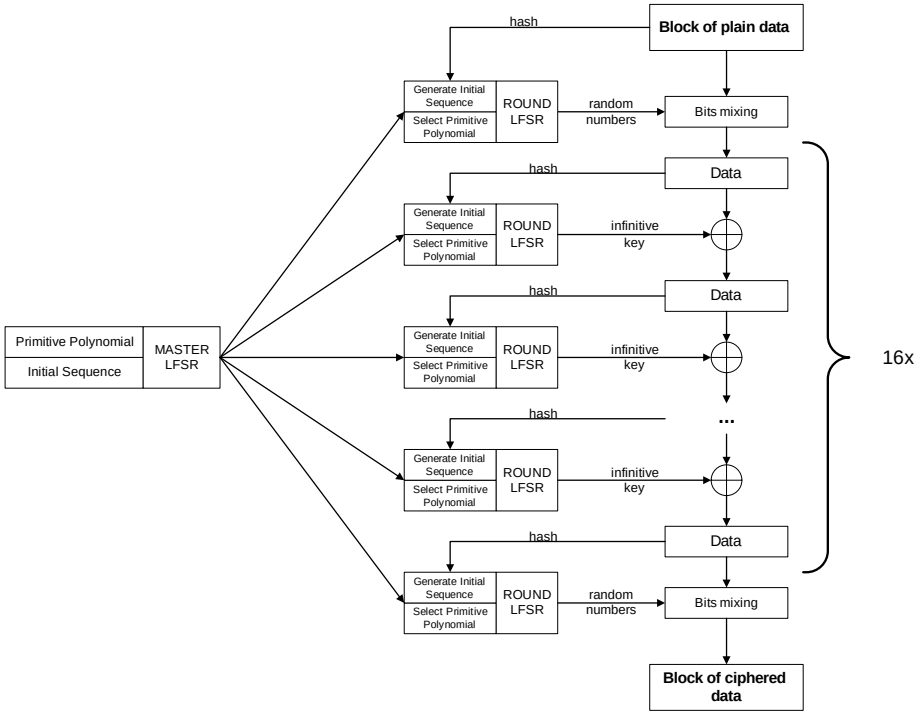

Fig. no. 4 - Ciphering in Karhu44.

The tap sequence is selected for each LFSR from the list at random. The random numbers generated by Master LFSR are used for doing selections, generation initial sequences for LFSR of rounds and in creating bits-mixing tables.

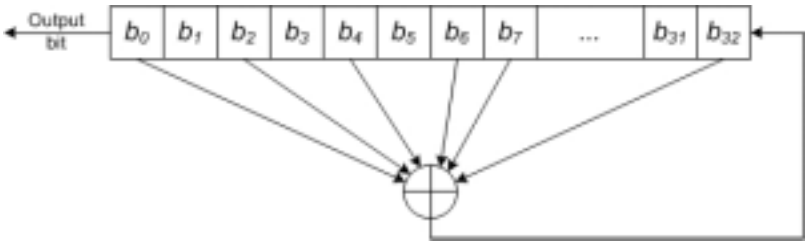

Fig. no. 5 - Linear feedback shift registers in Fibonacci configuration.

The length of registers of LFSR created based on that list varies from 50 up to 607 bits. This means that the minimum cycle of the register existing in this cryptosystem would be $2^{50} - 1 = 1{,}125899906842623 * 10^{15}$ what makes about 1024 TB. This is also the maximum length of a block can be processed.

The cryptographic key, supplied during creation, defines the initial state of the generator. After the initialisation generator is relaxed by generating at least that many bits how long is the register.

2.2 Generating of cryptographic key

Cryptographic key being used by KARHU44 algorithm consists of two data: primitive polynomial (taps sequence for LFSR) and initial sequence of the length that results from selected primitive polynomial.

The primitive polynomial is selected at random from the list of 229. List was generated based on data taken from (6). These sequences were evaluated against uniform distribution of bytes and bits generated. Selection criteria and methods of evaluating are presented in (15).

Initial sequence is generated based on ANSI standard X9.17 (16). The idea of this algorithm is presented on Fig. no. **5**. LSFR generators used in this algorithm are created in the way described in section 2.1. The initial state for those generators is defined at random with numbers generated by Master LFSR.

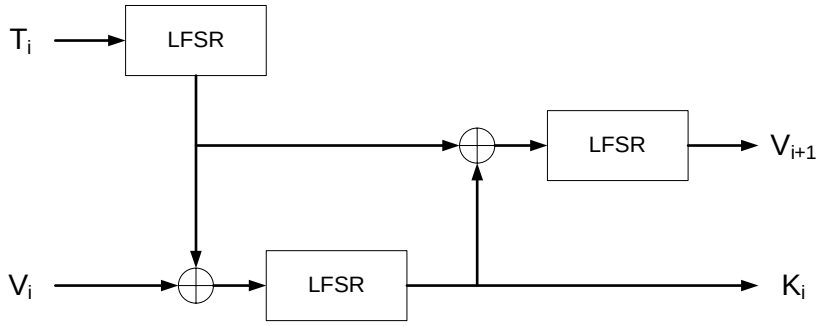

Fig. no. 5 - Cryptographic key generating standard ANSI X9.17

The time related element T_i is generated also by Master LFSR. Secret vector V_i is initialised with the session key (new value of this vector V_{i+1} overrides the old one V_i). The resulting stream K_i is the cryptographic key for the encoding / decoding round.

2.3 Level of security

There are several possible attacks on LFSR based stream ciphers. Good review of those methods can be found in (6). All of those attacks concern such ciphers that are made up of a few LFSRs but configuration (length and tap sequence) of component registers is constant. In case of the cipher KARHU44, the situation is different. The configuration of LFSR used for generating of encoding sequence is constant on one round only. I did not find a description of relevant cryptanalytic attack by now. Cryptanalysis of proposed cipher is the subject for further researches.

If we assume that, there are no practical attacks on the ciphering sequence now as well as that other typical cryptanalysis attacks on cipher cannot be applied because the cipher is having properties of one-time pad. The only possible attack is exhausting brute-force attack on session key. The length of session key is variable and at least 160 bits long. Taking into account works of LENSTRA & VERHEUL (5) we can estimate that data encoded by means of KARHU44 can be save for next 110 – 120 years. This range has been estimated based of extrapolation of data presented on Fig. no. ***1***.

The strong side of this algorithm is also fact that the key length is variable what makes brute force attack far harder. Random inclusion of tap sequence into the key makes brute force attack even harder.

Described by HŁOBAŻ (17) possibility of birthday attack on pseudo one-time pad ciphers does not concern this algorithm because hash functions are not use as means of ciphering.

In almost all block ciphers, the last block of a message being ciphered has to be padded with random sequence of bits. This is because it is happening very seldom that length of the message is congruent to block length. This makes good opportunity for cryptanalytical attack on the last block. This situation is not applied to KARHU44 because presented algorithm can process last block without padding what indeed improve security of proposed cipher.

2.4 Implementation

Presented algorithm was implemented as the program named KARHU44. The ciphering object is created with two data: maximum length of a block and session key.

Length of blocks being ciphered can vary between one byte and defined maximum length. Blocks which length is less than maximum one do not need to be padded.

Round key is generated with method described in section 2.2. Secret vector is initialised based on hash of the block data being processed. Time related element T_i is generated by Master LFSR.

Ciphering object can do encoding / decoding using all most popular modes: ECB, OFB, CFB and CBC. In all modes except ECB, the initial vector (IV) is set-up with random numbers obtained from Master LFSR. In future initial vector can be used to produce in parallel MAC as it was described in (17).

Program reads plain text and writes cipher in binary form.

KARHU44 project has been developed by means of Borland BCB 2007 development environment. Software has been design and coded in full according to object-oriented paradigm.

Table no. 2

Statistical evaluation of the KARHU44 algorithm.

Ciphering mode	Comparing average value of byte against theoretical one of 127,5 by means of zero hypothesis H_o: m = 127,5				Distribution of	
					bytes	bits
	Average (m)	sd (m)	t	Degree of freedom (df)	Chi2 (df = 255)	Chi2 (df = 1)
shakespeare.txt (4082 bytes)						
ECB	127,4	1,146	0,069	4081	272,132	0,344
OFB	127,8	1,159	0,259	4081	264,230	2,756
CFB	127,6	1,154	0,129	4081	209,669	0,000
CBC	128,9	1,159	1,218	4081	266,990	0,168
a.txt (4194305 bytes)						
ECB	127,5	0,036	0,398	4194304	258,733	0,025
OFB	127,5	0,036	1,022	4194304	277,598	0,080
CFB	127,5	0,036	0,719	4194304	254,836	0,004
CBC	127,6	0,036	2,289 *	4194304	221,492	2,256
SZ19_0063.JPG (2693558 bytes)						
ECB	127,6	0,045	1,384	2693557	240,722	0,001
OFB	127,5	0,045	0,854	2693557	275,718	1,456
CFB	127,5	0,045	0,356	2693557	238,269	0,264
CBC	127,5	0,045	0,081	2693557	247,676	0,216
mozart_sonata_p11.mp3 (3288592 bytes)						
ECB	127,5	0,041	0,469	3288591	273,757	1,715
OFB	127,5	0,041	0,381	3288591	268,877	0,595
CFB	127,6	0,041	1,451	3288591	272,900	0,133
CBC	127,5	0,041	0,063	3288591	268,207	2,295
thinking_in_cpp.pdf (4547906 bytes)						
ECB	127,5	0,035	1,251	4547905	240,731	2,137
OFB	127,5	0,035	0,185	4547905	251,201	0,098
CFB	127,5	0,035	0,554	4547905	258,755	0,344
CBC	127,5	0,035	1,033	4547905	278,509	0,061
publikacja.docx (1286951 bytes)						
ECB	127,6	0,065	0,819	1286950	250,891	0,720
OFB	127,5	0,065	0,762	1286950	236,938	0,453
CFB	127,5	0,065	0,291	1286950	233,032	0,431
CBC	127,4	0,065	0,774	1286950	241,148	0,004

3. Statistical evaluation

There are several criteria to evaluate ciphering algorithms. Vast discussion and review of methodologies can be found, for instance, in (8) (6). In addition, NIST publishes relevant recommendations on own website. The final criterion is always vulnerability for cryptanalysis. The primary criterion is distribution of bytes and bits in the cipher. Actually, the problem can be state as how far distribution of bytes and bits in the cipher differs from the uniform one. This criterion can be used for selection of algorithms or as a rough assessment. At current stage of this project, it has been decided to use three parameters that assess distribution of bytes in a cipher:

1. Chi^2 test for bytes distribution
2. Chi^2 test for bits distribution
3. Comparing average of bytes in cipher to the theoretical value of 127,5 by means of zero-hypothesis and t-Student distribution.
4. Compressing ratio of the cipher file

Table no. 3

Compression ratio of plain and ciphered contents

File name	File size [B]	Compression ratio				
		Plain file	Cipher in respective mode			
			ECB	OFB	CFB	CBC
shakespeare.txt	4 082	58,0 %	0,0 %	0,0 %	0,0 %	0,0 %
a.txt	4 194 305	99,999 %	0,0 %	0,0 %	0,0 %	0,0 %
SZ19_0063.JPG	2 693 558	0,9 %	0,0 %	0,0 %	0,0 %	0,0 %
mozart_sonata_p11.mp3	3 288 592	1,5 %	0,0 %	0,0 %	0,0 %	0,0 %
thinking_in_cpp.pdf	4 547 906	15,1 %	0,0 %	0,0 %	0,0 %	0,0 %
publikacja.docx	1 286 951	2,3 %	0,0 %	0,0 %	0,0 %	0,0 %

Points (1), (2) and (3) are standard approach to evaluation of numbers distribution. Criterion (4) needs some consideration.

A file can be packed if it contains on various positions similar sequences of characters. Those sequences are coded what makes resulting file compressed. In file containing ideal random values, it is impossible to find such sequences. Therefore, the sequence of random values cannot be compressed. For example, file A.TXT contains 4 MB of character A (see Table no. 3) and it is compressed in 99,999 % while file SZ19_0063.JPG is compressed with ratio 0,9 % only, because it is already compressed as JPEG file. In other words, if a file contains random numbers its compression ratio should be close to zero percent. To carry out described test program WinRAR version 3.71 working in “BEST” compression mode has been used.

For the test, six files have been used:

1. SHAKESPEARE.TXT — Plain text in English. File created by means of Notepad.
2. A.TXT — Plain text containing 4 MB of character A without characters like.
3. SZ19_0063.JPG — The picture in JPEG format.
4. PUBLIKACJA.DOCX — MS Word document containing text in English and pictures.
5. MOZART_SONATA_P11.MP3 — Audio file in MP3 format.
6. THINKING_IN_CPP.PDF — Document in PDF format.

All files have been ciphered in modes listed in section 2.4. The statistical evaluation of ciphers is presented in Table no. 2. Distribution of bytes in file A.TXT containing 4 MB of character A as well as in file containing cipher obtained in CFB mode is presented on Fig. no. **6**. The similar examples for audio data are presented on Fig. no. **7**.

Results of tests from (1) to (3) proof – with probability 95% – that the distribution of bytes in all tested cipher files is uniform (see Table no. 2). Average of value of bytes of cipher obtained in CBC mode from file A.TXT slightly exceed theoretical value of 127,5. Taking into account results of both Chi2 test it might be assumed that distribution of value of bytes should be uniform in this case too. From Table no. 2 the conclusion that ciphering mode does not have any impact on the distribution of byte values can be drawn.

Moreover, based on results presented in Table no. 3 test (4) proofs that all cipher files contain random values.

4. Discussion

On the base of statistical evaluation we can draw the conclusion that the proposed cipher KARHU44 generates sequences of random numbers having uniform distribution irrespective of the kind of clear data. The proposed cipher dissolves data well in all tested ciphering modes. Moreover, there is no difference between ciphers obtained in these modes. Results for the A.TXT (see also Fig. no. **6**) data file are especially valuable because this is the one of the basic statistical tests in selection of ciphers.

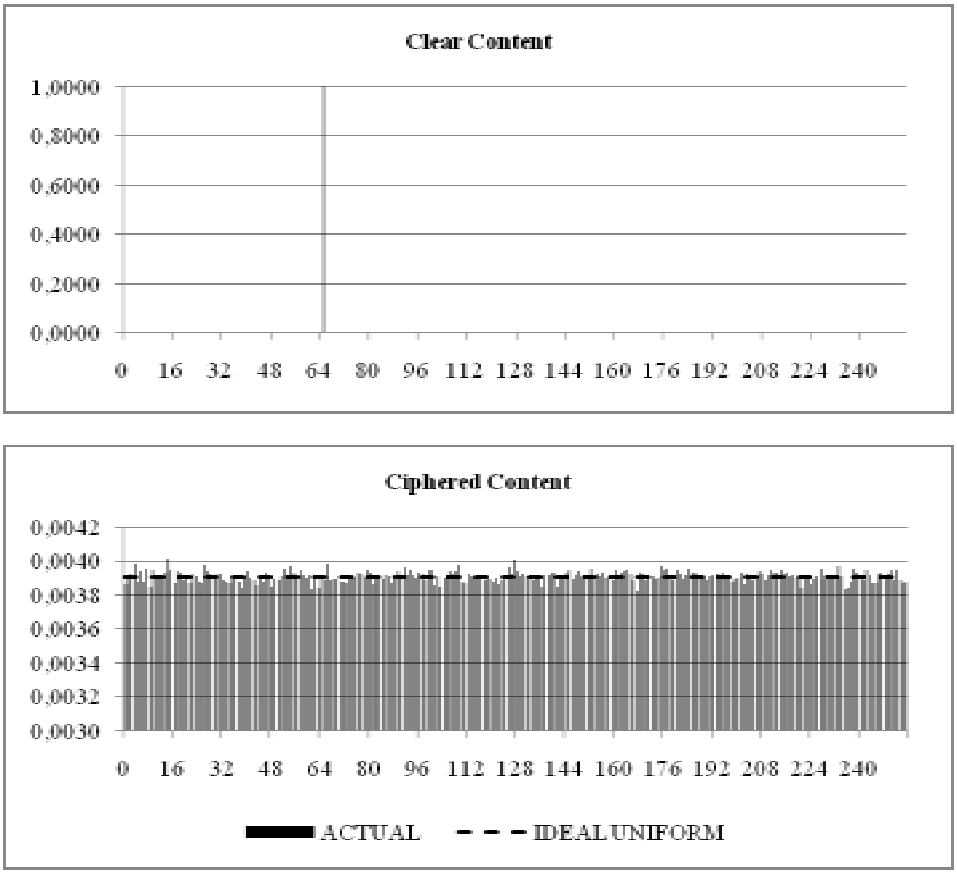

Fig. no. 6 - Distribution of bytes in A.XXX files (CFB mode).

The construction of KARHU44 is simple compared to other ciphers; therefore its software implementation is secure. Blocks of information with a maximum size of 1024 TB can be processed

safely, but shorter blocks such as 256-512 bytes long are recommended. KARHU44 can assure security of stored data for more than the next one hundred years.

The current results of the test (this paper includes selected examples only) allow to recommend the proposed algorithm to be used for long-term protection of data and content stored on any type of electronic media. The cryptographic properties of this algorithm make cracking attempts unfeasible within the time the prospective hacker would like to posses the targeted content.

Taking into account the results of the primary test, the KARHU44 algorithm can be presented for public discussion and testing. All comments and test results are welcome and will be taken into account in further works. Suggestions of further research can be formulated as follows:

- To make cryptanalysis of the cipher and to investigate the use of other than LFSR random number generators;
- To make time profiling of the current code, to optimise the code, and to investigate the possibility of hardware implementation;
- To investigate the possibility of parallel MAC calculation during ciphering in all modes, except ECB;
- Further investigation of the behaviour of the cipher for content protection and data archiving.

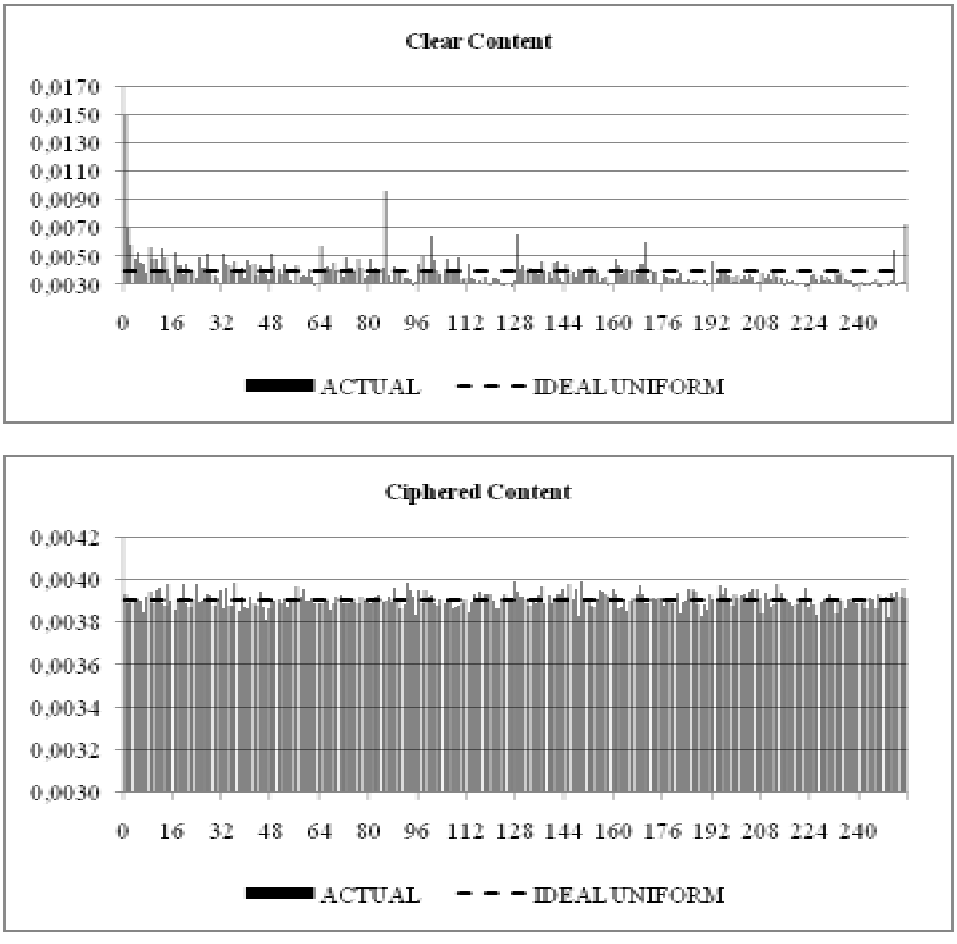

Fig. no. 7 - Distribution of bytes in MOZART_SONATA_P11.XXX files (CFB mode).

References

1. La cryptographie militaire. Kerckhoffs, Auguste. 1883, Journal des sciences militaires, Vol. IX, pp. 161-191.

2. Kerckhoffs, Auguste. 1883, Journal des sciences militaires, Vol. IX, pp. 5-83.

3. Raymond, Eric S. The Cathedral & the Bazaar. s.l. : O'Reilly, 1999. ISBN 1565927249.

4. Gordon, Lawrence A, et al. 2004 CSI/FBI Computer Crime and Security Survey. San Francisco : Computer Security Institute, 2005.
5. Selecting Cryptographic Key Sizes. Lenstra, Arjen K and Verheul, Eric R. 2001, Journal of Cryptology, p. 14.
6. Schneier, Bruce. Applied Cryptography. 2nd Edition. s.l. : John Wiley & Sons, 1966. ISBN 0471117099.
7. Rubin, F. Rubin F., One-time pad cryptography. [Online] 13 01 1977. [Cited: 05 07 2005.] http://www.contestcen.com/crypt005.htm.
8. Bauer, Friedrich L. Sekrety kryptografii. 3rd Edition. Katowice : Helion, 2002. ISBN 8371799606.
9. Martinez, Sylvain. BUGS Dynamic Cryptography Algorithm. [Online] 19 11 2000. [Cited: 04 07 2005.] http://www.encryptsolutions.com/english/info/doc/bugs_tech_main.html.
10. Ritter, Terry. New Encryption Technologies for Communications Designers. [Online] 01 05 2005. [Cited: 03 07 2005.] http://www.ciphersbyritter.com/CRYPHTML.HTM.
11. Yarrow-160: Notes on the Design and Analysis of the Yarrow Cryptographic Pseudorandom Number Generator. Kelsey, John, Schneier, Bruce and Ferguson, Niels. [ed.] H Heyes and C Adams. s.l. : Springer-Verlag, 1999. Lecture Notes in Computer Science. p. 1758.
12. Cryptanalytic Attacks on Pseudorandom Number Generators. Kelsey, John, et al. [ed.] S Vaudenay. Berlin / Heidelberg : Springer-Verlag, 1998. Lecture Notes in Computer Science. pp. 168-188. ISBN 978-3-540-64265-7.
13. Szewczyk, Grzegorz. Construction and generation of cryptographic keys for ciphers based on LFSR. In print..
14. Reinhold, Edward M, Nievergelt, Jurg and Deo, Nsrshing. Algorytmy kombinatoryczne. Warsaw : PWN, 1985. ISBN 8301051701.
15. Szewczyk, Grzegorz. Valuation and selection of tap sequences for LFSR based random number generator. In print..
16. Генератор псевдослучайных чисел ANSI X9.17. [Online] 17 08 2002. [Cited: 03 07 2005.] http://aforge.ibd.lv/?35.
17. Hłobaż, Artur. Bezpieczeństwo transmisji danych w systemach pomiarowych. Łódź University of Technology. Łódź, Poland : s.n., 2008. PhD Thesis.